Data Min Knowl Disc (2007) 14:99–129
DOI 10.1007/s10618-006-0049-3

Compression-based data mining of sequential data

Eamonn Keogh · Stefano Lonardi · Chotirat Ann Ratanamahatana · Li Wei · Sang-Hee Lee · John Handley

Received: 15 December 2005/Accepted: 2 May 2006 /
Published online: 26 January 2007

Abstract The vast majority of data mining algorithms require the setting of many input parameters. The dangers of working with parameter-laden algorithms are twofold. First, incorrect settings may cause an algorithm to fail in finding the true patterns. Second, a perhaps more insidious problem is that the algorithm may report spurious patterns that do not really exist, or greatly overestimate the significance of the reported patterns. This is especially likely when the user fails to understand the role of parameters in the data mining

Responsible editor: Johannes Gehrke.

E. Keogh (✉) S. Lonardi · L. Wei
Department of Computer Science and Engineering, University of California, Riverside, CA 92521, USA
e-mail: eamonn@cs.ucr.edu

S. Lonardi
e-mail: stelo@cs.ucr.edu

L. Wei
e-mail: wli@cs.ucr.edu

C. A. Ratanamahatana
Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand
e-mail: ann@cp.eng.chula.ac.th

S. H. Lee
Department of Anthropology, University of California, Riverside, CA 92521, USA
e-mail: shlee@ucr.edu

J. Handley
Xerox Innovation Group, Xerox Corporation, 800 Phillips Road, MS 128-25E, Webster, New York 14580-9701, USA
e-mail: jhandley@xeroxlabs.com

process. Data mining algorithms should have as few parameters as possible. A parameter-light algorithm would limit our ability to impose our prejudices, expectations, and presumptions on the problem at hand, and would let the data itself *speak* to us. In this work, we show that recent results in bioinformatics, learning, and computational theory hold great promise for a parameter-light data-mining paradigm. The results are strongly connected to Kolmogorov complexity theory. However, as a practical matter, they can be implemented using any off-the-shelf compression algorithm with the addition of just a dozen lines of code. We will show that this approach is competitive or superior to many of the state-of-the-art approaches in anomaly/interestingness detection, classification, and clustering with empirical tests on time series/DNA/text/XML/video datasets. As a further evidence of the advantages of our method, we will demonstrate its effectiveness to solve a real world classification problem in recommending printing services and products.

Keywords Kolmogorov complexity · Parameter-free data mining · Anomaly detection · Clustering

1 Introduction

Most data mining algorithms require the setting of many input parameters. There are many dangers of working with parameter-laden algorithms. We may fail to find true patterns because of poorly chosen parameter settings. A perhaps more insidious problem is that we may find patterns that do not exist (Keogh et al. 2003), or greatly overestimate the significance of a pattern because of a failure to understand the role of parameter searching in the data mining process (Domingos 1998; Elkan 2001). In addition, as we will show, it can be very difficult to compare the results across methods or even to reproduce the results of heavily parameterized algorithms.

Data mining algorithm should ideally have as few parameters as possible. A parameter-free algorithm prevents us from imposing our prejudices and presumptions on the problem at hand, and lets the data itself *speak* to us.

In this work, we consider a data mining paradigm based on compression. The work is motivated by results in bioinformatics, learning, and computational theory that are not well known outside those communities. As we will demonstrate here, our approach allows parameter-free or parameter-light solutions to many classic data mining tasks, including clustering, classification, and anomaly detection.

Our approach has the following advantages, which we will empirically demonstrate with extensive experiments:

(1) It allows true *exploratory* data mining, rather than forcing us to impose our presumptions on the data.
(2) The accuracy of our approach can be greatly superior to those of parameter-laden algorithms, even if we allow these algorithms to search exhaustively over their parameter spaces.

(3) Our approach is based on compression as its cornerstone, and compression algorithms are typically space and time efficient. As a consequence, our method can be much more efficient than other algorithms, in some cases by three or four orders of magnitude.
(4) Many parameterized algorithms require the data to be in a special format. For concreteness, consider time series data mining (Ge and Smyth 2000; Keogh and Ka setty 2002). Here, the Euclidean distance requires that the dimensionality of two instances being compared is exactly the same, and Dynamic Time Warping (DTW) is not defined if a single data point is missing (Ratanamahatana and Keogh 2004). In contrast, our approach works for time series of different lengths, sampling rates, dimensionalities, with missing values, etc.

In this work, we decided to take the unusual step of reproducing our entire actual *code*, rather than just the pseudocode. There are two reasons for doing this. First, free access to the actual code combined with our policy of making all data freely available allows independent confirmation of our results. Second, it reinforces our claim that our methods are very simple to implement.

The rest of the paper is organized as follows. In Sect. 2, we discuss the results in bioinformatics and computational theory that motivate this work. In Sect. 3, we consider the minor changes and extensions necessary to extend these results to the classic data mining tasks of anomaly/interestingness detection, classification, and clustering. Section 4 sees an exhaustive empirical comparison, in which we compare dozens of algorithms to our approach, on dozens of datasets from several domains, including time series, video, DNA, and text. In Sect. 5, we show preliminary results from a real world application of our technique in the problem in recommending high-end printing services and products. Finally, in Sect. 6, we discuss many avenues for possible extensions.

2 BACKGROUND AND RELATED WORK

We begin this section by arguing that a research contribution made by a parameter-laden algorithm can be difficult to evaluate. We review some background material on Kolmogorov complexity, which motivates the parameter-free Compression-based Dissimilarity Measure (CDM), the technique at the heart of this paper.

2.1 The perils of parameter-laden algorithms

A recent paper in a top-tier journal introduced a new machine-learning framework and noted that it "…abounds with parameters that *can* be tuned" (our emphasis). It is surely more accurate to state that the approach has parameters that *must* be tuned. When surveying the literature on this topic, we noted that while there are many techniques for automatically tuning parameters, many of

these techniques themselves have parameters, possibly resulting in an infinite regression.

An additional problem of parameter-laden algorithms is that they make it difficult to reproduce published experimental results and to understand the contribution of a proposed algorithm.

A recently published paper introduced a new time series distance measure. The algorithm requires the setting of two parameters, and the authors are to be commended for showing the results of the cross-product: 16 by four possible parameter choices. Of the 64 settings, 11 are slightly better than DTW, and the authors conclude that their approach is superior to DTW. However, the authors did not test over different parameters for DTW, and DTW does allow a single parameter, the maximum temporal distortion (the "*warping window*" (Ratanamahatana and Keogh 2004). The authors kindly provided us with the exact data they used in the experiment, and we reproduced the experiment, this time allowing a search over DTW's single parameter. We discovered that over a wide range of parameter choices, DTW produces a near perfect accuracy, outperforming all 64 choices of the proposed algorithm.

Although the above is only one anecdotal piece of evidence, it does help make the following point. It is very difficult to evaluate the contribution of papers that introduce a parameter-laden algorithm.

In the case above, the authors' commendable decision to make their data public allows the community to discover that DTW is probably a better distance measure, but only at the expense of some effort on the readers' behalf. In general, the potential asymmetry in parameter tuning effort effectively prevents us from evaluating the contribution of many papers. Here, the problem is compounded by the fact that the authors themselves created the dataset in question. Creating a dataset may be regarded as a form of meta-parameter tuning, since we do not generally know if the very first dataset created was used in the paper, or many datasets were created and only the most satisfactory one was used. In any case, there clearly are problems in setting parameters (training) and reporting results (testing) on the *same* dataset (Salzberg 1997). In the field of neural networks, Flexer (1996) noted that 93% of the papers did just that. While no such statistics are published for data mining, an informal survey suggests a similar problem may exist here. In Sect. 4.2.2, we will empirically reinforce this point by showing that in the context of anomaly detection, parameter-laden algorithms can have their parameters tuned to achieve excellent performance on one dataset, but completely fail to generalize to a new but very similar dataset.

Before leaving this section, it would be remiss of us not to note that many papers by the authors of this manuscript also feature algorithms that have (too) many parameters. Indeed, the frustration of using such algorithms is one inspiration for this work.

2.2 Kolmogorov complexity

The proposed method is based on the concept of Kolmogorov complexity, a measure of randomness of strings based on their information content. It was proposed by Kolmogorov in 1965 to quantify the randomness of strings and other objects in an objective and absolute manner.

The Kolmogorov complexity $K(x)$ of a string x is defined as the length of the shortest program capable of producing x on a universal computer — such as a Turing machine. Different programing languages will give rise to distinct values of $K(x)$, but one can prove that the differences are only up to a fixed additive constant. Intuitively, $K(x)$ is the minimal quantity of information required to generate x by an algorithm.

Hereafter, we will follow the notation of Li et al. (2003), which was the main inspiration of this work. The conditional Kolmogorov complexity $K(x|y)$ of x to y is defined as the length of the shortest program that computes x when y is given as an auxiliary input to the program. The function $K(xy)$ is the length of the shortest program that outputs y concatenated to x.

In Li et al. (2001), the authors consider the distance between two strings, x and y, defined as

$$d_k(x, y) = \frac{K(x|y) + K(y|x)}{K(xy)} \tag{1}$$

which satisfies the triangle inequality, up to a small error term. A more mathematically precise distance was proposed in Li et al. (2003). Kolmogorov complexity is without a doubt the ultimate lower bound among all measures of information content. Unfortunately, it cannot be computed in the general case Li and Vitanyi (1997). As a consequence, one must approximate this distance.

It is easy to realize that universal compression algorithms give an upper bound to the Kolmogorov complexity. In fact, $K(x)$ is the best compression that one could possibly achieve for the text string x. Given a data compression algorithm, we define $C(x)$ as the size of the compressed size of x and $C(x|y)$ as the compression achieved by first training the compression on y, and then compressing x. For example, if the compressor is based on a textual substitution method, one could build the dictionary on y, and then use that dictionary to compress x.

We can approximate (1) by the following distance measure

$$d_c(x, y) = \frac{C(x|y) + C(y|x)}{C(xy)} \tag{2}$$

The better the compression algorithm, the better the approximation of d_c for d_k is.

In Li et al. (2003) have shown that d_c is a similarity metric and can be successfully applied to clustering DNA and text. However, the measure would require hacking the chosen compression algorithm in order to obtain $C(x|y)$ and $C(y|x)$.

We therefore decided to simplify the distance even further. In the next section, we will show that a simpler measure can be just as effective. A comparative analysis of several compression-based distances has been recently carried out in Sculley and Brodley (2006).

The idea of using data compression to classify sequences over finite alphabets is not new. For example, in the early days of computational biology, lossless compression was routinely used to classify and analyze DNA sequences. We refer to, e.g., Allison et al. (2000), Baronchelli et al. (2005), Farach et al. (1995), Frank et al. (2000), Gatlin (1972), Kennel (2004), Kit (1998), Loewenstern et al. (1995), Loewenstern and Yianilos (1999), Mahnoney (2003, Unpublished), Needham and Dowe (2001), Segen (1990), and Teahan et al. (2000), and references therein for a sampler of the rich literature existing on this subject.

More recently, Benedetto et al. (2002) have shown how to use a compression-based measure to classify fifty languages. The paper was featured in several scientific (and less-scientific) journals, including Nature, Science, and Wired. It has also generated some controversies (see, e.g., Goodman 2002, Unpublished).

The use of data compression to classify sequences is also tightly connected with the Minimum Description Length (MDL) and Minimum Message Length (MML) principles. The MDL principle, introduced in the late 1970s by Rissanen (1978), holds that the best explanation, given a limited set of observed data, is the one that can be described as succinctly as possible, or equivalently that delivers the greatest compression of the data. In other words, the more we are capable to compress the data, the better the model we are building to "explain" the data, and the more we are learning about the regularities underlying the data. The MDL is not the first information-theoretic approach to machine learning. In fact, (Wallace and Boulton 1968) proposed in 1968 a related concept called Minimum Message Length (MML) (see also Needham and Dowe (2001)).

The MDL/MML principle has generated an extensive body of literature in the data mining community. Without pretending to be exhaustive, example of applications ranges from inferring Mehta et al. (1995) and pruning Papadimitriou et al. (2005) decision trees, to discovery of cross-associations (Chakrabarti et al. 2004) or highly repetitive subgraphs in a labeled graph (Cook and Holder 2000), all the way to spatial data mining (Papadimitriou et al. 2005).

2.3 Compression-based dissimilarity measure

Given two strings, x and y, we define the CDM as follows

$$\mathrm{CDM}(x,y) = \frac{C(xy)}{C(x) + C(y)}. \tag{3}$$

The CDM dissimilarity is close to 1 when x and y are not related, and smaller than one if x and y are related. The smaller the $\mathrm{CDM}(x,y)$, the more closely related x and y are. Note that $\mathrm{CDM}(x,x)$ is not zero.

Table 1 Compression-based dissimilarity measure (CDM)

```
function Dist = CDM(A,B)
save A.txtA–ASCII                          % Savevariable A as A.txt
zip('A.zip', 'A.txt');                     % Compress A.txt
A_file = dir('A.zip');                     % Get file information
save B.txt B –ASCII                        % Save variable B as B.txt
zip('B.zip', 'B.txt');                     % Compress B.txt
B_file = dir('B.zip');                     % Get file information
A_n_B = [A; B];                            % Concatenate A and B
save A_n_B.txt A_n_B –ASCII                % Save A_n_B.txt
zip('A_n_B.zip', 'A_n_B.txt');             % Compress A_n_B.txt
A_n_B_file = dir('A_n_B.zip');             % Get file information
                                           % Return CDM dissimilarity
dist = A_n_B_file.bytes / (A_file.bytes + B_file.bytes);
```

The dissimilarity measure can be easily implemented. The entire Matlab code is shown in Table 1.

The inputs are the two matrices to be compared. These matrices can be time series, DNA strings, images, natural language text, midi representations of music, etc. The algorithm begins by saving the two objects to disk, compressing them, and obtaining file information. The next step is to concatenate the two objects (A_n_B = [A; B]); the resulting matrix is also saved, compressed, and the file information is retrieved. At this point, we are almost done; we simply return the size of the compressed concatenation over the size of the sum of the two individual compressed files.

One could argue that CDM also has several parameters. In fact, CDM depends on the choice of the specific compressor (gzip, compress, bzip2, etc.), and on the compression parameters (for example, the sliding window size in gzip). But because we are trying to get the best approximation of the Kolmogorov complexity, one should just choose the *best* combination of compression tool and compression parameters for the data. There is, in fact, no freedom in the choice to be made. We simply run these compression algorithms on the data to be classified and choose the one that gives the highest compression. For all the experiments in this work, we simply use the default compression algorithm used in Matlab, with the default parameters. The sole exception was for DNA, where a compression algorithm optimized for the four-letter alphabet is known to produce superior results.

2.4 Choosing the representation of the data

As we noted earlier, the only objective in CDM is to obtain good compression. There are several ways to achieve this goal. First, one should evaluate several compressors. If we have domain knowledge about the data under study, and specific compressors are available for that type of data, we should use one of those. For example, if we are clustering DNA, we should consider a compression algorithm optimized for compressing DNA [see, e.g., (Benedetto et al. 2002)].

A	B	C
0.13812500000000	0.51250000000000	0.49561523437690
0.04875000000000	0.50000000000000	0.49604248046834
0.10375000000000	0.50000000000000	0.49653076171875
0.17875000000000	0.47562500000000	0.49706481933594
0.24093750000000	0.45125000000000	0.49750732421875
0.29875000000000	0.45125000000000	0.49808715820312
0.37000000000000	0.47656250000000	0.49875854492187
0.48375000000000	0.50000000000000	0.49939941406230
0.55593750000000	0.48281250000000	0.50007080078125
0.64625000000000	0.48468750000000	0.50062011718750
0.70125000000000	0.46937500000000	0.50123046875826

Fig. 1 The first ten data points of three ECGs

There is another way we can help improve the compression; we can simply ensure that the data to be compared is in a format that can be readily and meaningfully compressed. Consider the following example; Fig. 1 shows the first ten data points of three Electrocardiograms from PhysioNet (Goldberger et al. 2000) represented in textual form.

It happens to be the case that sequences **A** and **C** are both from patients with supraventricular escape beats. If we are allowed to see a few hundred additional data points from these sequences, we can correctly group the sequences ((**A,C**),**B**) by eye, or with simple Euclidean distance.

Unfortunately, CDM may have difficulties with these datasets. The problem is that although all sequences are stored with 16-digit precision, sequences **A** and **B** were actually recorded with 8-digit precision and automatically converted by the Rdsamp-O-Matic tool (Goldberger et al. 2000). Note that, to CDM, **A** and **B** may have great similarity, because the many occurrences of 00000000's in both **A** and **B** will compress even better in each other's company. In this case, CDM is finding true similarity between these two sequences, but it is a trivial *formatting* similarity, and not a meaningful measure of the *structure* of the heartbeats. Similar remarks can be made for other formatting conventions and hardware limitations; for example, one sensor's number-rounding policy might produce a surfeit of numbers ending with "5."

Before explaining our simple solution to this problem, we want to emphasize that CDM is extremely robust to it. For example, all the anomalies detected in Sect. 4.2 can be easily discovered on the original data. However, addressing this problem allows us to successfully apply CDM on much smaller datasets.

A simple solution to the problem noted above is to convert the data into a discrete format, with a small alphabet size. In this case, every part of the representation contributes about the same amount of information about the shape of the time series. This opens the question of which symbolic representation of time series to use. In this work, we use the SAX (Symbolic Aggregate ApproXimation) representation of Lin et al. (2003). This representation has been shown to produce competitive results for classifying and clustering time series, which suggests that it preserves meaningful information from the original data. Furthermore, the code is freely available from the authors' website. While SAX

does allow parameters, we use the parameterless version for all experiments here.

Similar remarks can be made for other data types. For example, when clustering WebPages, we may wish to strip out the HTML tags first. Imagine we are trying to cluster WebPages based on authorship, and it happens that some of the WebPages are graphic intensive. The irrelevant (for this task) similarity of having many occurrences of "<IMG SRC…>" may dominate the overall similarity. In Section 5, we encounter a very similar problem in our real world application of CDM to clustering case logs.

3 Parameter-Free data mining

Most data mining algorithms, including classification (Domingos 1998), clustering (Gavrilov et al. 2000; Kalpakis et al. 2001; Keogh et al. 2003), anomaly/interestingness detection(Shahabi et al. 2000; Ma and Perkins 2003; Christen and Goiser 2005), reoccurring pattern (motif) discovery, similarly search (Wallace and Boulton 1968), etc., use some form of similarity/dissimilarity measure as a subroutine. We will consider just the first three tasks in this work.

3.1 Clustering

As CDM is a dissimilarity measure, we can simply use it directly in most standard clustering algorithms. For some partitional algorithms (Elkan 2003), it is necessary to define the concept of cluster "center." While we believe that we can achieve this by extending the definition of CDM, or embedding it into a metric space (Faloutsos and Lin 1995), for simplicity here, we will confine our attention to hierarchical clustering.

3.2 Anomaly detection

The task of finding anomalies in data has been an area of active research, which has long attracted the attention of researchers in biology, physics, astronomy, and statistics, in addition to the more recent work by the data mining community (Shahabi et al. 2000; Ma and Perkins 2003; Christen and Goiser 2005). While the word "anomaly" implies that a radically different subsection of the data has been detected, we may actually be interested in more subtle deviations in the data, as reflected by some of the synonyms for anomaly detection, interestingness/deviation/surprise/novelty detection, etc.

For true parameter-free anomaly detection, we can use a divide-and-conquer algorithm as shown in Table 2. The algorithm works as follows: Both the left and right halves of the entire sequence being examined are compared to the entire sequence using the CDM dissimilarity measure. The intuition is that the side containing the most unusual section will be less similar to the global sequence than the other half. Having identified the most interesting side, we can recur-

Table 2 Parameter-free anomaly detection algorithm

```
function loc_of_anomaly = kolmogorov_anomaly(data)
loc_of_anomaly = 1;
while size(data,1) > 2
  left_dist = CDM(data(1:floor(end/2),:),data);
  right_dist = CDM(data(ceil(end/2):end,:),data);
  if left_dist < right_dist
      loc_of_anomaly = loc_of_anomaly + size(data,1) / 2;
      data = data(ceil(end/2):end,:);
  else
      data = data(1:floor(end/2),:);
  end
end
```

sively repeat the above, repeatedly dividing the most interesting section until we can no longer divide the sequence.

This twelve-line algorithm appears trivial, yet as we shall see in Sect. 4.2, it outperforms four state-of-the-art anomaly detection algorithms on a wide variety of real and synthetic problems. The algorithm has another important advantage; it can handle both single dimensional anomaly detection and multi-dimensional anomaly detection without changing a single line of code. We will demonstrate this ability in Sect. 4.2.3

While the algorithm above easily detects the anomalies in all the datasets described in Sect. 4, there are two simple ways to greatly improve it further. The first is to use the SAX representation when working with time series, as discussed in Sect. 2.4. The second is to introduce a simple and intuitive way to set a single parameter. The algorithm in Table 2 allows several potential weaknesses for the sake of simplicity. Firstly, it assumed a single anomaly in the dataset. Second, in the first few iterations, the measure needs to note the difference a small anomaly makes, even when masked by a large amount of surrounding normal data. A simple solution to these problems is to set a parameter W, for number of windows. We can divide the input sequence into W contiguous sections, and assign the anomaly value of the ith window as CDM(W_i, *data*). In other words, we simply measure how well a small local section can match the global sequence. Table 3 shows the entire Matlab code to achieve this.

Setting this parameter is not too burdensome for many problems. For example of the ECG dataset discussed in Sect. 4.2, we found that we could find the objectively correct answer if the size of the window ranged anywhere from a 1/4 heartbeat length to four heartbeats. For clarity, we call this slight variation Window Comparison Anomaly Detection (WCAD).

3.3 Classification

Because CDM is a dissimilarity measure, we can trivially use it with a lazy-learning scheme. For simplicity, in this work, we will only consider the one-nearest-

Table 3 Window comparison anomaly detection algorithm

```
function kolmogorov_anomaly_WCAD (data)
W = 16;        % the number of windows, this is the only user defined parameter
anomaly_values = [];
 for i = 1 : floor(size(data,1)/W) : (size(data,1) - floor(size(data,1)/W) )
   temp = data;
   sub = data(i : i + floor(size(data,1)/W),:);
   temp(i : i + floor(size(data,1)/W),:) = [];
   rest = temp;
   anomaly_values = [ anomaly_values ; CDM(sub, rest)];
 end;
[val,location] = max(anomaly_values);
disp(['The anomaly begins at location ', int2str((location-1)*(size(data,1)/W)) ])
```

neighbor algorithm. Generally speaking, lazy learners using non-metric proximity measures are forced to examine the entire dataset. However, one can use an embedding technique such as FASTMAP (Faloutsos and Lin 1995) to map the objects into a metric space, thus allowing indexing and faster classification. For simplicity, we disregard this possibility in this work.

4 Empirical Evaluation

While this section shows the results of many experiments, it is actually only a subset of the experiments conducted for this research project. We encourage the interested reader to consult (Keogh 2004) for additional examples.

4.1 Clustering

While CDM can work with most clustering techniques, here we confine our attention to hierarchal clustering, since it lends itself to immediate visual confirmation.

4.1.1 Clustering time series

In order to perform convincing experiments, we wanted to test our algorithm against all reasonable alternatives. However, lack of space prevents us from referencing, much less explaining them. So, we re-implemented *every* time series distance/dissimilarity/similarity measure that has appeared in the last decade in any of the following conferences: SIGKDD, SIGMOD, ICDM, ICDE, VLDB, ICML, SSDB, PKDD, and PAKDD. In total, we implemented 51 such measures, including the ten mentioned in Keogh and Ka setty (2002) and the eight variations mentioned in Gavrilov et al. (2000). A list of all these measures can be found in appendix [*A*]. For fairness, we should note that many of these measures are designed to deal with short time series, and make no claim about their ability to handle longer time series. In addition to the above, we considered

the classic Euclidean distance, DTW, the L_1 metric, the L_{inf} metric, and the Longest Common Subsequence (LCSS), all of which are more than a decade old. Some of these (Euclidean and the other L_p metrics) are parameter free. For measures that require a single parameter, we did an exhaustive search for the best parameter. For measures requiring more than one parameters (one method required seven!), we spent one hour of CPU time searching for the best parameters using a genetic algorithm and independently spent 1 h searching manually for the best parameters. We then considered only the better of the two.

For our first experiment, we examined the UCR Time Series Archive (Keogh and Folias 2002) for datasets that come in pairs. For example, in the Foetal-ECG dataset, there are two time series, *thoracic* and *abdominal*, and in the Dryer dataset, there are two time series, *hot gas exhaust* and *fuel flow rate*. We were able to identify 18 such pairs, from a diverse collection of time series covering the domains of finance, science, medicine, industry, etc. Although our method is able to deal with time series of different lengths, we truncated all time series to length 1,000 to allow comparisons to methods that require equal length time series.

While the correct hierarchical clustering at the top of the tree is somewhat subjective, at the lower level of the tree, we would hope to find a single bifurcation separating each pair in the dataset. Our metric, Q, for the quality of clustering is therefore the number of such correct bifurcations divided by 18, the number of datasets. For a perfect clustering, $Q = 1$, and because the number of dendrograms of 36 objects is greater than 3×10^{49}, for a random clustering, we would expect $Q \approx 0$.

For each measure, we clustered using single linkage, complete linkage, group average linkage, and wards methods, and reported only the best performing result. Figure 2 shows the resulting dendrogram for our approach.

Our approach achieved a perfect clustering, with $Q = 1$. Although the higher level clustering is subjective, here too our approach seems to do very well. For example, the appearance of the *Evaporator* and *Furnace* datasets in the same subtree is quite intuitive, and similar remarks can be made for the two *Video* datasets and the two *MotorCurrent*datasets.

More than $^3/_4$ of the other approaches we tested scored $Q = 0$. Several of the parameter-laden algorithms suffer from the following limitation. Although their parameters could be carefully tuned to do well on one type of data, say the relatively smooth *MotorCurrent* datasets, they achieve poor performance on the more noisy datasets like *Balloon*. We could then tune the parameters to do better on the noisy datasets, but immediately lose discriminatory power on the smooth data.

The only measures performing significantly better than random were the following. *Euclidean distance* had $Q = 0.27$. *DTW* was able to achieve $Q = 0.33$ after careful adjustment of its single parameter. *The Hidden Markov* Model approach of Ge and Smyth (2000) had $Q = 0$ using the original piecewise linear approximation of the time series. However, when using the SAX representation, its score jumped to $Q = 0.33$. The *LPC Cepstra* approach of Kalpakis et al.

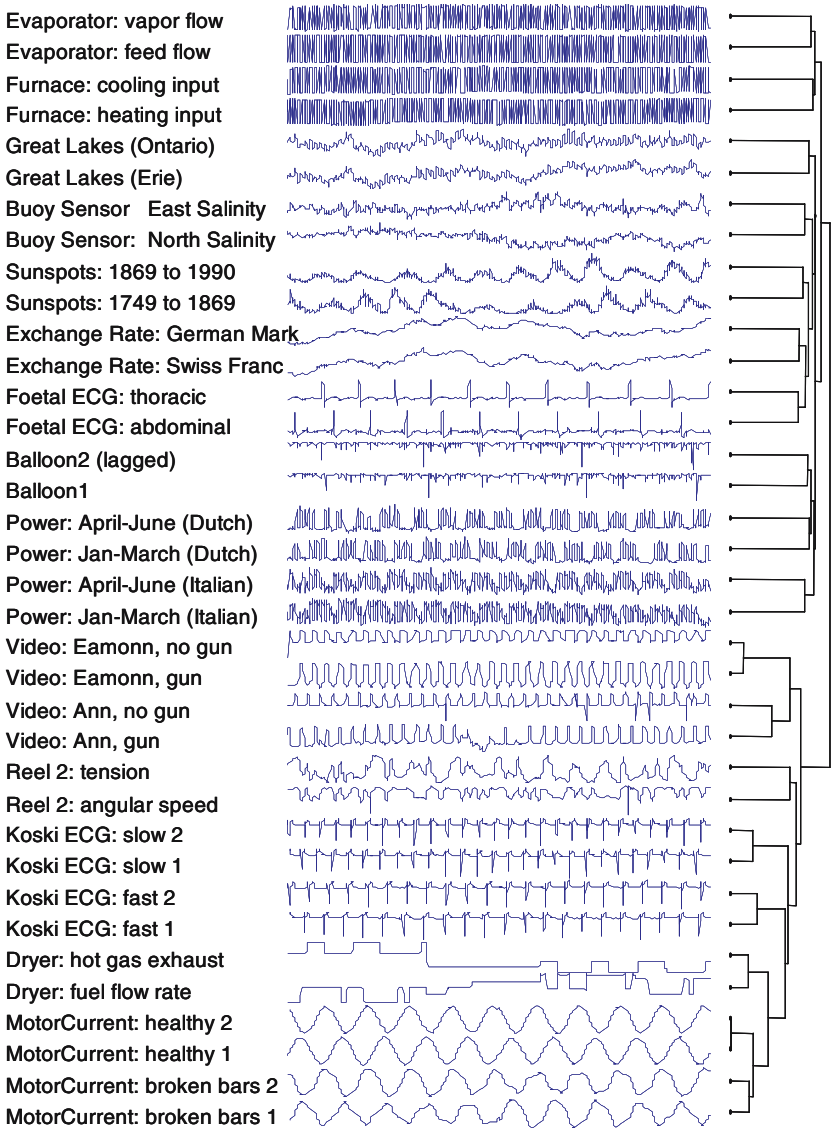

Fig. 2 Thirty-six time series (in 18 pairs) clustered using the approach proposed in this paper

(2001) and the similar *Autocorrelation* method of [55] both had $Q = 0.16$. *LCSS* had $Q = 0.33$.

Our first experiment measured the quality of the clustering only at the leaf level of the dendrogram. We also designed a simple experiment to test the quality of clustering at a higher level. We randomly extracted ten subsequences of length 2,000 from two ECG databases. For this problem, the clustering at the leaf level is subjective. However, the first bifurcation of the tree should divide

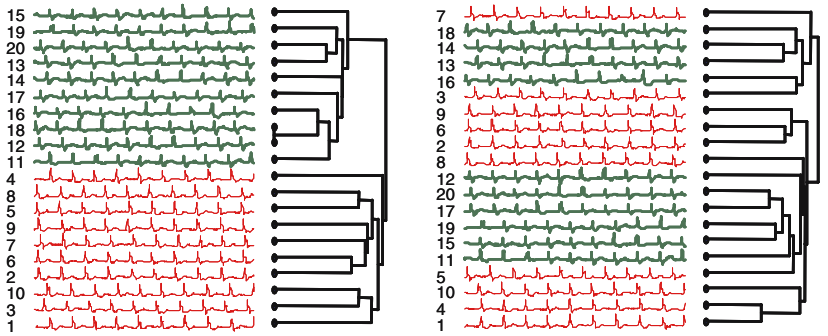

Fig. 3 Two clusterings on samples from two records from the MIT-BIH Arrhythmia Database (*Left*) Our approach (*Right*) Euclidean distance

the data into the two classes. (the probability of this happening by chance is only 1 in 524,288.) Figure 3 shows the two best clusterings obtained.

In a sense, our exhaustive comparison to other similarity methods was unfair to many of them, which can only measure the similarity of a few local shapes, rather than the higher-level structural similarity required.

The following "trick" improved the results of most of the algorithms on both problems above. To compare two time series A and B of length n, we can extract a subsequence of length s from A, and compare it to every location in B, then record the closest match as the overall distance between A and B. Although this does help the majority of the similarity measures, we note the following. It adds a new (and highly sensitive) parameter to set and increases the time complexity by a factor of $O(n^2)$, and even after making this optimization, none of the competing similarity measures come close to the performance of CDM.

We also conducted an experiment in clustering robotic sensors. The Sony AIBO is a small, quadruped robot that comes equipped with a tri-axial accelerometer. This accelerometer returns data at a rate of 125 Hz. The robotics group at Carnegie Mellon University (CMU) (who graciously donated the data) made the robot walk on two different surfaces.

- *Field*: The carpet used on a robot soccer field. It resembles a thin layer of felt over an approximately 1cm foam backing.
- *Cement*: A cement floor.

The CMU group is interested in the problem of clustering the robots experience. The robot records three sensors (X, Y, Z). However, for simplicity, we only consider the X-axis here. For this problem, the differences between the two classes are very subtle, unlike the two problems considered above; we could not visually distinguish between the two classes, even after very careful inspection on a high-resolution monitor. Figure 4 shows the result of clustering ten 8s robot experiences (5 from each class).

On this problem, LPC Cepstra approach of Gavrilov et al. (2000) and the Autocorrelation method of Quinlan and Rivest (1989) also performed much better than random, but only CDM repeatedly produced perfect clusterings.

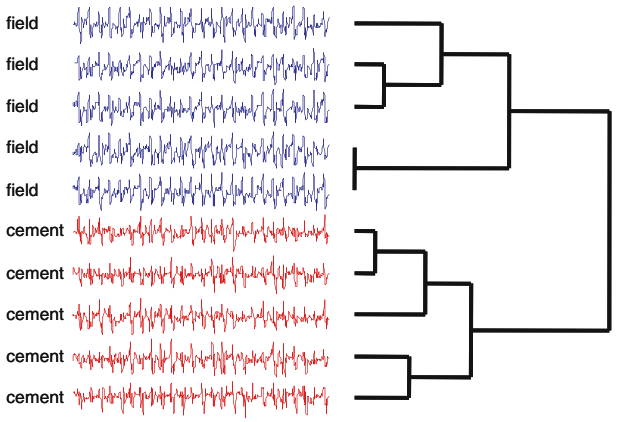

Fig. 4 Robot sensors clustered using CDM

Finally, while the results of these experiments are very promising for our approach, some fraction of the success could be attributed to luck. To preempt this possibility, we conducted many additional experiments, with essentially identical results. These experiments are documented in Keogh (2004).

4.1.2 Clustering DNA and text

As a test of our ability to cluster text, we began by conducting experiments on DNA strings. We took the first 16,300 symbols from the mitochondrial DNA of twelve primates and one "outlier" species, and hierarchically clustered them. A similar strategy was used in Li et al. (2003) on a different set of organisms. One of the current authors, Lee, is an expert in primate evolution. She has compared the tree to the state-of-the-art consensus in the primatology literature and notes that while some of the branch *lengths* are still the subject of controversy, the topography of the tree is correct (Figure 5).

We want to note that using a compressor optimized for DNA Benedetto et al. (2002) was essential here. A standard dictionary-based compressor similar to gzip, would give less meaningful distances.

We conducted additional experiments with a more diverse collection of animals; in every case, the clustering agreed with the current consensus on evolutionary history Keogh (2004).

We also examined natural language text. A similar experiment is reported in Benedetto et al. (2002). Here, we began by clustering the text of various countries' Yahoo portals. We only considered the first 1,615 characters, the size of the smallest webpage (excluding white spaces). Figure 6 (*left*) shows the clustering result. Note that the first bifurcation correctly divides the tree into *Germanic* and *Romance* languages. While we striped out all HTML tags for this experiment, we found that leaving them in made little difference, presumably because they are more or less equally frequent across languages.

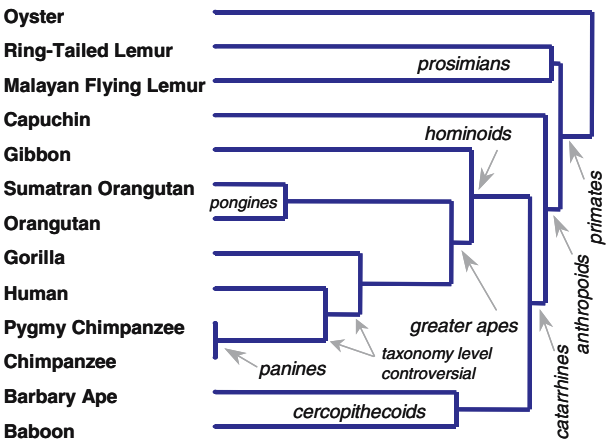

Fig. 5 The clustering achieved by our approach on 16,300 symbols from the mitochondrial DNA of 12 primates, and one "outlier" species

Surprisingly, the clustering shown is much better than that achieved by the ubiquitous cosine similarity measure used at the word level. In retrospect, this is hardly surprising.

Consider the following English, Norwegian, and Danish words taken from the Yahoo portals:

English: {England, *information*, addresses}
Norwegian: {Storbritannia, *informasjon*, adressebok}
Danish: {Storbritannien, *informationer*, adressekartotek}

Because there is no single word in common to all (even after applying Porters algorithm), the three vectors are completely orthogonal to each other in vector

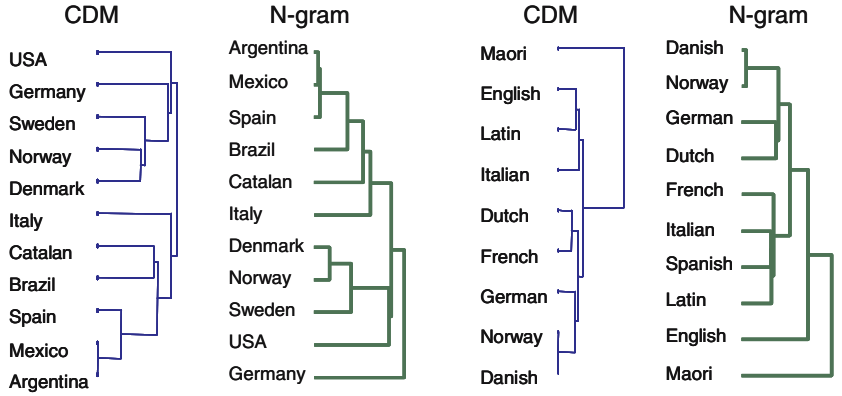

Fig. 6 (*Left*) The clustering achieved by our approach and N-Gram cosine measure on the text from various Yahoo portals (January 15, 2004). The smallest webpage had 1,615 characters, excluding white spaces. (*Right*) The clustering achieved by our approach on the text from the first 50 chapters of Genesis. The smallest file had 132,307 characters, excluding white spaces. Maori, a Malayo–Polynesian language, is clearly identified as an "outlier"

space. However, any human inspection of the text is likely to correctly conclude that Norwegian and Danish are much more similar to each other than they are to English. Our approach can leverage off the same cues by finding repeated structure within and across texts. To be fairer to the cosine measure, we also tried an N-Gram approach at the character level. We tested all values of N from 1 to 5, using all linkages methods from the following set {Wards, single, complete, group}. The results are reported in Figure 6. While the *best* of these results are produces an intuitive clustering, other settings do not.

We tried a similar experiment with text from various translations of the first 50 chapters of the bible, this time including what one would expect to be an outlier, the Maori language of the indigenous people of New Zealand. As shown in Figure 6 *(right)* the clustering is correct, except for an inclusion of French in the *Germanic* subtree.

Once again, we reiterate the following disclaimer. We are not suggesting that our method replace the vector space model for indexing text, or a linguistic aware method for tracing the evolution of languages. Our point is simply to show that given a dataset which we know nothing about, we can expect our approach to produce reasonable results that can be a starting point for future study.

4.2 Anomaly detection

Although our approach can be used to find anomalies in text, video, images, and other data sources, we will confine our attention here to time series, since this domain has attracted the most attention in the data mining community and readily lends itself to visual confirmation.

For all the problems shown below, we can objectively discover the anomaly using the simple algorithm in Table 2. However, that algorithm only tells us the *location* of the anomaly, without telling us anything about the relative *strength* of the anomaly. For this reason, we use the Window Comparison Anomaly Detection (WCAD) variation discussed in Sect. 3.2 and shown in Table 3. This slight variation allows us to determine the relative strength of the anomaly, which we can visualize by mapping onto the line's thickness. The level of novelty is simply the value of CDM (sub, rest), where "sub" is the local subsection, and "rest" is the entire rest of the time series (excluding sub), i.e., the value calculated in line 9 of Table 3.

As noted in Sect. 3.2, WCAD does have one simple parameter to set, which is W, the approximate size of the window we expect to find anomalies in. In these experiments, we only count an experiment as a success for CDM if the first window size we choose finds the anomaly, and if window sizes four times and one quarter as large can also find the anomaly.

We consider four rival techniques. Here, we list them, and state the number of parameters each requires in parentheses. We refer the interested reader to the original papers for more details. We compared our approach to the Support Vector Machine (SVM) based approach of Ma and Perkins (2003) (6),

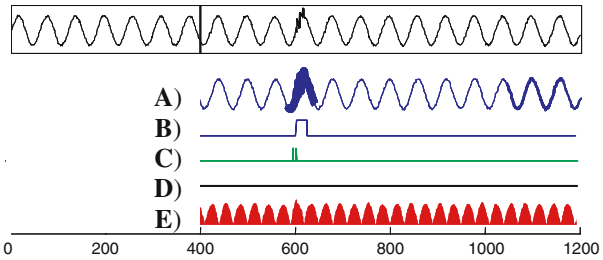

Fig. 7 A comparison of five novelty detection algorithms on the *synthetic sine* problem of Ma and Perkins (2003). The first 400 data points are used as training data. An "event" is embedded at time point 600. **a** The approach proposed in this work; the thickness of the line encodes the level of novelty. **b** SVM. **c** IMM. **d** AR. **e** TSA

the Immunology (IMM) inspired approach of Christen and Goiser (2005) (5), The Association Rule (AR) based approach of Yairi et al. (2001) (5), and the TSA-tree Wavelet based approach of Shahabi et al. (2000) (3). As before, for each experiment, we spent one hour of CPU time, and one hour of human time trying to find the best parameters and only reported the best results.

4.2.1 A simple normalizing experiment

We begin our experiments with a simple sanity check, repeating the *noisy sine* problem of Ma and Perkins (2003). Figure 7 shows the results.

Our approach easily finds the novelty, as did SVM with careful parameter tuning. The IMM algorithm is stochastic, but was able to find the novelty in the majority of runs. We were simply unable to make the AR approach work. Finally, TSA does peak for the novelty, although its discriminatory power appears weak.

The ability of our approach to simply match the prowess of SVM and IMM on this problem may not seem like much of an achievement, even though we did it orders of magnitude faster and without setting any parameters. However, the real utility of our approach becomes evident when we see how the algorithms generalize, or when we move from toy problems to real world problems. We consider both cases below.

4.2.2 Generalizability experiment

To illustrate the dangers of working with parameter-laden algorithms, we examined a generalization of the last experiment. As shown in Figure 8, the training data remains the same. However, in the test data, we changed the period of the sine wave by a barely perceptible 5%, and added a much more obvious "anomaly", by replacing a half of a sine wave with its absolute value. To be fair, we modified our algorithm to only use the training data as reference. To do this, we compare each subsection of the test data, not with the rest of the

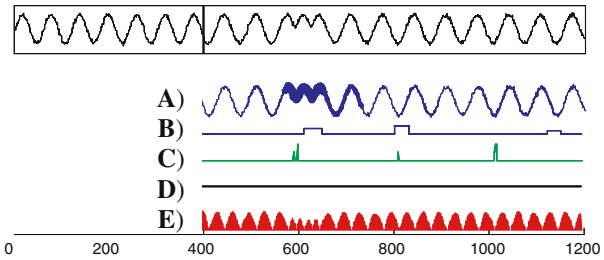

Fig. 8 A comparison of five novelty detection algorithms on a generalization of the *synthetic sine* problem. The first 400 data points are used as training data. In the rest of the time series, the period of the sine wave was changed by 5%, and one half of a sine wave was replaced by its absolute value. **a** The approach proposed in this work; the thickness of the line encodes the level of novelty. b SVM. c IMM. d AR. e TSA

training data, but with all the training data. This is achieved by replacing line 9 of Table 3 with anomaly_values = [anomaly_values ; CDM(sub, training_data)];.

The results are telling; while our algorithm easily finds the new anomaly, SVM and IMM discover more important "anomalies" elsewhere. It may be argued that the very slight change of period *is the anomaly*, and these algorithms did the right thing. However, we get a similar inability to generalize if we instead slightly change the amplitude of the sine waves, or if we add (or remove!) more uniform noise or make any other innocuous changes, including ones that are imperceptible to the human eye.

In case the preceding example was a coincidentally unfortunate dataset for the other approaches, we conducted many other similar experiments. And since creating our own dataset opens the possibly of data bias (Keogh and Ka setty 2002), we considered datasets created by others. We were fortunate enough to obtain a set of 20 time series anomaly detection benchmark problems from the Aerospace Corp. A subset of the data is shown in Fig. 9.

The TSA algorithm easily discovered the anomaly in the time series *L*-1*j*, but not the other two time series. We found that both SVM and IMM could have their parameters tuned to find the anomaly on any individual one of the

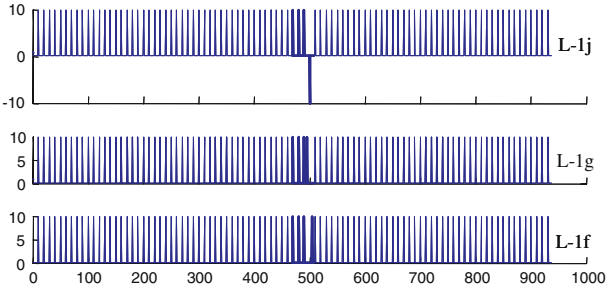

Fig. 9 The results of applying our algorithm to (a subset of) a collection of anomaly detection benchmark datasets from the Aerospace Corp. The thickness of the line encodes the level of novelty. In every case, an anomaly was inserted beginning at time point 500

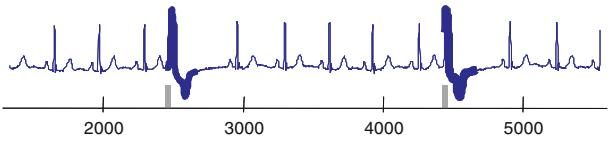

Fig. 10 A small excerpt from dataset 118e06 from the MIT-BIH Noise Stress Test Database. The full dataset is 21,600 data points long. Here, we show only a subsection containing the two most interesting events detected by our algorithm (*the bolder* the line, the more interesting the subsequence). The *gray* markers are independent annotations by a cardiologist indicating Premature Ventricular Contractions

three sequences, but once the parameters were tuned on one dataset, they did not generalize to the other two problems.

The objective of these experiments is to reinforce the main point of this work. Given the large number of parameters to fit, it is nearly impossible to avoid overfitting.

Before leaving this section we would like to briefly relate an anecdote as a further support for our approach. For the above problem, we wrote a simple Matlab script to read in the 20 datasets, run our anomaly detection algorithm, and confirm that the most anomalous section was discovered within 25 points of 500. After successfully testing our approach, we modified the script to consider the other approaches but found that it always crashed when working with dataset *L*-1*s*. After some careful debugging, we discovered that the artificial anomaly in this sequence is some missing data points, which are encoded in Matlab as the special character "NaN". While none of the other algorithms are defined for missing values (hence the crashing), and are not trivially extendible, our approach was robust enough not to crash, and to find the right answer.

4.2.3 Real-world anomaly detection

We examined annotated datasets from the MIT-BIH Noise Stress Test Database. For the algorithms which need a training/test split, we gave them 1/3 of the dataset which had been annotated as normal. We then asked the algorithms to examine the rest of the data to find the most interesting events, comparing the results to the cardiologists' annotations. Figure 10 shows the result of one such experiment. Note that only a small excerpt from the full dataset is shown.

We only illustrate the performance of our approach in Fig. 10 because all the other approaches produced results that were objectively (per the cardiologists' annotations) and subjectively incorrect, in spite of careful parameter tuning.

Our final example illustrates the flexibility of our approach. None of the approaches for anomaly detection in time series in the literature are defined for multidimensional time series[1], in spite of an increasing general interest in multidimensional time series (Vlachos et al. 2003). However, we can consider

[1] This includes the four rival approaches considered here (Quinlan and Rivest 1989; Segen 1990; Loewenstern et al. 1995; Christen and Goiser 2005). While the TSA-Wavelet approach was extended to 2D, this extension is for *spatial* mining.

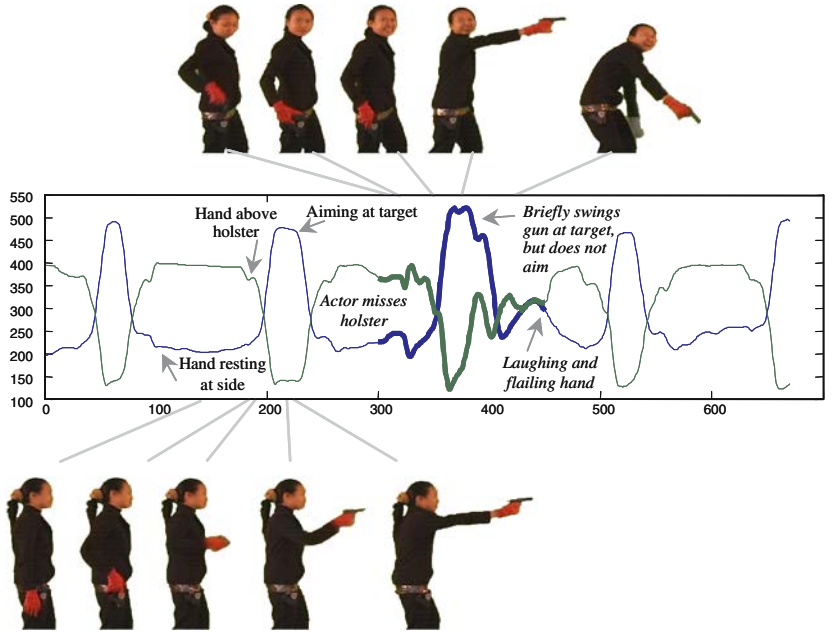

Fig. 11 (*Bottom*) A typical video snippet from the Gun video is mapped onto a two-dimensional time series (*Center*) by tracking the actor's right hand. While the vast majority of the dataset looks approximately like the first 200 data points, the section from about 300 to 450 looks somewhat different, and was singled out by our anomaly detection algorithm. Examining the original video (*Top*), we discovered the cause of the anomaly

multidimensional time series without changing a single line of code. In order to have some strawman to compare to, for each of the four competing methods, we adapted them so that although they work on each dimension individually, we linearly combined their results into a single measure of novelty.

We experimented on a 2D time series that was collected for a different purpose (in particular, a classification problem (Ratanamahatana and Keogh 2004). The 2D time series was extracted from a video of an actor performing various actions with and without a replica gun. Figure 11 (*bottom*) illustrates a typical sequence. The actor draws a replica gun from a hip mounted holster, aims it at a target, and returns it to the holster.

Watching the video, we discovered that at about 10 s into the shoot, the actor misses the holster when returning the gun. An off-camera (inaudible) remark is made, the actor looks toward the video technician, and convulses with laughter. At one point (frame 385), she is literally bent double with laughter. This is the only interesting event in the dataset, and our approach easily finds it. The other techniques returned results that do not seem truly anomalous, given a careful inspection of both the time series and the original video.

We have not considered time efficiency as a metric in these experiments, because we cannot guarantee that our implementations of the rival approaches are as efficient as they might be, given careful optimization. However, our

approach is reasonably fast, requiring less than ten seconds (on a 2.65 GHz machine) to process a million data points.

4.3 Classification

In this section, we illustrate the utility of CDM for classification with the following simple experiment. We use the following similarity measures on four datasets (Two each from two databases: ECG and Gun) and measure their error rates:

- Euclidean distance (Keogh and Ka setty 2002).
- Dynamic time warping (DTW). Here, we exhaustively test all values of its single parameter (warping window size [47]) and report only the best result, and
- Compression-Based Dissimilarity Measure (CDM)

For brevity, in this section, we only compare CDM with DTW and Euclidean Distance metric, since it has been shown in Keogh and Ka setty (2002) that many of the more complex similarity measures proposed in other work have higher error rates than a simple Euclidean Distance metric.

The ECG datasets are four-class problem derived from BIDMC Congestive Heart Failure Database (Goldberger et al. 2000) of four patients. Since this original database contains two ECG signals, we separate each signal and create two datasets of one-dimensional time series in the following way. Each instance of 3,200 contiguous data points (about 20 heartbeats) of each signal is randomly extracted from each long ECG signals of each patient. Twenty instances are extracted from each class (patient), giving eighty total instances for each dataset.

The Gun datasets are time-series datasets extracted from video sequences of two actors either aiming a gun or simply pointing at a target (Ratanamahatana and Keogh 2004) (see also, Figure 11). We randomly extract twenty instances of 1,000 contiguous data points (about seven reps) from each of the following long time series:

(A) Actor 1 with gun
(B) Actor 1 without gun (point)
(C) Actor 2 with gun
(D) Actor 2 without gun (point)

The first dataset is a two-class problem of differentiating Actor 1 from Actor 2 — (A+B) vs. (C+D). The second dataset is a four-class problem of differentiating each of the acts independently — A vs. B vs. C vs. D. In total, each dataset contains eighty instances. Some samples from both databases are shown in Fig. 12.

We measure the error rates on each dataset, using the one-nearest-neighbor with 'leaving-one-out' evaluation method. The lower bounding technique noted in Ratanamahatana and Keogh (2004) is also integrated in all the DTW

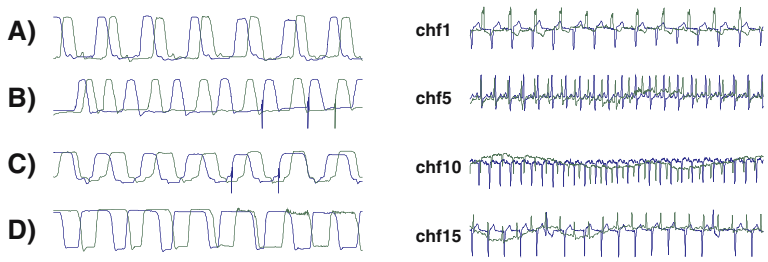

Fig. 12 Some extracted time series from the gun datasets (*left*) and the ECG (*sig.1*) dataset (*right*)

Table 4 Classification error rates (%) for all four datasets

	Euclidean (%)	DTW (best unif. window) (%)	CDM(%)
ECG: signal 1	42.25	16.25	6.25
ECG: signal 2	47.50	11.25	7.50
Gun: two classes	5.00	0.00	0.00
Gun: four classes	37.50	12.5	5.00

calculations to help achieve speedup. The experimental results are summarized in Table 4.

In all four datasets discussed above, Euclidean distance is extremely fast, yet inaccurate. DTW with the best uniform window size greatly reduces the error rates, but took several orders of magnitude longer. However, CDM outperforms both Euclidean and DTW in all datasets. Even though CDM is slower than Euclidean distance, it is much faster than the highly optimized DTW.

We do not give exact times here since CDM is implemented in the relatively lethargic Matlab, whereas DTW is implemented in highly optimized C++. Nevertheless, even if we excluded the time taken to find search over DTW's single (and sensitive, see Ratanamahatana and Keogh (2004)) parameter, CDM is still about 25 times faster than DTW.

5 A real world case study

Xerox Corporation sells printer systems and software to production print shops. A production print shop is a business that provides printed hard-copy to end customers. These may be internal organizations in a large corporation that produces manuals for its own use, for example, or a neighborhood printing business. During the presale stage, sales analysts need to recommend a set of printers and software to the print shops according to their printing requirements. A consistent set of printers and software that meets a customer's needs is called a workflow configuration. Because there is a bewildering array of software and printer configurations available for purchase, sales analysts use a web-based

workflow configuration tool to elicit customer requirements and provide automated recommendations. The workflow configuration tool comprises a dynamic questionnaire of the customer's printing requirements, such as locations, products (e.g., direct mailings, brochures, catalogs), color specification, etc. The questionnaire is dynamic in the sense that which questions to be presented next depends on how previous questions were answered. Question responses include yes/no, free text, multiple and single (radio-button) selections. After the requirements are filled in, a rule-based workflow auto-generation module intelligently generates a set of capability-equivalent workflow configurations based upon the product and workflow knowledge patterns in a knowledge base. The user then chooses one workflow that is most suitable to his or her requirements. User requirements and the generated workflows are recorded by the tool as case logs in XML format. Figure 13 shows a sample segment of a case log.

Xerox is primarily interested in whether the high-level workflow can be detected from the questions and responses to validate the rule-based workflow configuration system. Also, the tool can be augmented by using historical data to classify new cases. An incoming case is classified in two simple steps.

- First, group existing case logs into some clusters and characterize each cluster with its representative workflow configurations;
- Second, when a new case comes, classify it using one nearest neighbor and recommend the representative workflows of that cluster.

The first step is essentially a clustering problem. However, case logs represent a challenge for clustering because there are no obviously meaningful distance measures. The usual vector space models for text do not apply here because many of the responses are simply yes–no or true–false. Transformed term fre-

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<GUIJspbean>
 <GUIQuestionnaireQuestionnaireVector>
  …
  <GUIQuestionnaireLocalizableMessage>Printing Application Types (select all that apply)
  </GUIQuestionnaireLocalizableMessage>
  <GUIQuestionnaireSelectMultipleChoice isSelected="false">
                    <GUIQuestionnaireLocalizableMessage>General Commercial / Annual Reports
   </GUIQuestionnaireLocalizableMessage>
    </GUIQuestionnaireSelectMultipleChoice>
  <GUIQuestionnaireSelectMultipleChoice isSelected="false">
                  <GUIQuestionnaireLocalizableMessage>Books</GUIQuestionnaireLocalizableMessage>
   </GUIQuestionnaireSelectMultipleChoice>
 …
 </GUIQuestionnaireQuestionnaireVector>
 <GUIJspbeanAllWorkflows>
  <OntologyWorkflowName>POD::Color Split::iGen3/Nuvera::iWay</OntologyWorkflowName>
  …
 </GUIJspbeanAllWorkflows>
</GUIJspbean>
```

Fig. 13 A sample case log (abbreviated for brevity)

quencies, however, are poor representations of case log content. String (or graph) edit distance would seem to be another appropriate choice. However, the logs, even with many XML tags stripped, are 10-24 kilobytes in size, posing computational problems. Due to the inapplicability of the classic distance measurements, we decided to adopt CDM for our clustering task.

Note that the case logs are in XML format, relatively unstructured, and can have widely varying lengths. The XML tags are for describing and formatting purposes. They are *not* useful for our clustering task. Indeed, they are even misleading: two XML documents may have the same XML tags but totally different attributes and values. However, because there are so many tags and the tags are long, they occupy a large fraction of the document. Thus, CDM may regard these two documents as very similar to each other. In this case, CDM is finding true similarity between these two XML documents, but it is a trivial *structure* similarity, and not a meaningful measure of the requirements of the users. This situation is analogous to the observation made for the time series data in Fig. 1. Based on this observation, we strip out the XML tags from the case logs and only keep the attribute-value pairs (in plain text format).

For simplicity, we confine our attention to hierarchical clustering. We consider all the three most common linkage metrics, *single linkage*, *complete linkage*, and *average linkage*. We convert the hierarchical clustering to k partitional clusters by the well known technique of "cutting" the dendrogram into subtrees at the k –1 internal nodes closest to the root. We compare our results to those achieved by a highly tuned in-house approach based on probabilistic latent semantic analysis (PLSA) to see how our approach performs compared to a state-of-the-art probabilistic document clustering approach (Gaussier et al. 2002).

We randomly selected 200 cases from January 13, 2005 to July 12, 2005. The relatively small size of the dataset is due to the need to have human experts annotate each record to obtain the ground truth.

We discovered that CDM-based hierarchical clustering methods are able to distinguish workflow themes much better than PLSA. For example, when $k = 3$, the CDM hierarchical clustering methods discover three pure workflow themes, *POD and Bookprinting*, *Personalized and Transactional/Promotional*, and *no workflow recommendation* (this is the cluster where no workflow has been generated even though users have answered the questionnaire). Figure 14 shows the dendrogram of the clusters generated by average linkage. In contrast, when $k = 3$, PLSA only identified two workflow themes, mixing in the *no workflow recommendation* with the other two classes.

The most surprising result is that the clustering method with CDM was able to detect a previously unknown class when the user exits the tool prematurely. These are (now) called "*no workflow*". These cases cannot be detected purely by length and they cannot be easily detected by inspection. There were 23 cases of "*no workflow*", ten of which were misclassified nearly uniformly among the other classes: three as "*Book*," three as "*POD*", two as "*Transactional/Promotional*", and two as "*Personalized*". Also, CDM with hierarchical agglomerative clustering was able to detect the close relationship between the Book Printing and POD workflows — customers buy similar product configurations for these.

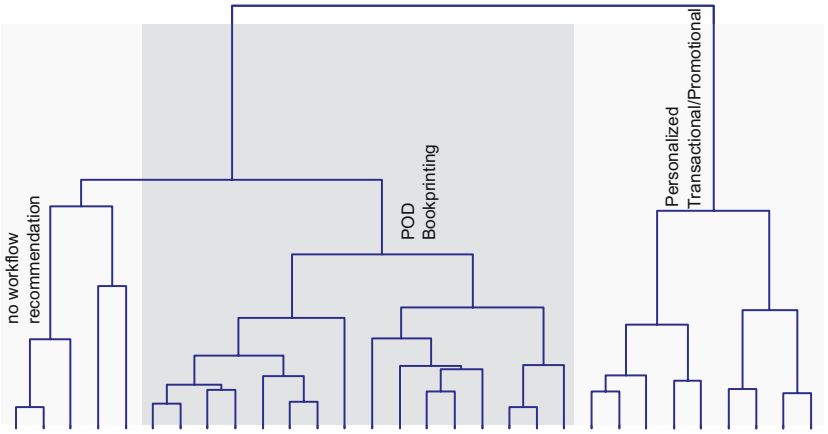

Fig. 14 The dendrogram of 200 cases obtained by CDM average linkage. The dendrogram is truncated at level 30 for visual clarity. Note that below level 3, the dendrogram separates the data into three pure classes

Table 5 Class distribution of the 200 case logs

Class	Num of case logs
Bookprinting	47
No workflow recommendation	23
Personalized	33
POD	70
Transactional/promotional	27

Having clustered the case logs, it is possible for us to classify new case logs. For simplicity, we consider the one-nearest–neighbor classification algorithm here. The idea is straightforward: when a new case log comes, we compute the CDM between it and all the logs in the case log database. Then we assign the new case log to the cluster that its closest neighbor belongs to. One may notice that after inserting a few case logs, the clusters assignment of the whole database may change. To handle this, our solution is to use a classic idea in the database community, *lazy update* (Ferrandina et al. 1994). The basic idea is that, when a new case log is added into the database, we simply use the current distance matrix to classify it, and wait for an opportunity to recompute the distance matrix and the clusters. For example, we can perform all the updates at a scheduled time when we are unlikely to compete with human users for CPU time.

Again, we use our 200 case logs having XML tags being stripped out and workflows being removed. Since we have the workflow information in the original logs, we know that these 200 case logs fall into five classes, as shown in Table 5.

We measure the error rates obtained by different clustering methods, using the one-nearest–neighbor with "leaving-one-out" evaluation method. The experimental results are summarized in Table 6 with boldface figures indicating

Table 6 Classification error rates (%) for all methods

	Complete linkage(%)	Average linkage(%)	Single linkage(%)	PLSA(%)
$k = 3$	5	4%	11	26
$k = 4$	5	4%	10.5	25.5
$k = 5$	5	4%	10.5	16.5
$k = 6$	4	4%	10.5	7

the winning method. In all the methods and cluster number settings, *average linkage* always has the lowest error rates. *Complete linkage* is slightly worse, but is still much better than *single linkage* and PLSA.

6 Conclusions and future work

In this work, we argued that data mining algorithms with many parameters are burdensome to use, and make it difficult to compare results across different methods. We further showed empirically that at least in the case of anomaly detection, parameter-laden algorithms are particularly vulnerable to overfitting. Sometimes they achieve perfect accuracy on one dataset, and then completely fail to generalize to other very similar datasets (Elkan 2001).

As a step towards mitigating these problems, we showed that parameter-free or parameter-light algorithms can compete with or outperform parameter-laden algorithms on a wide variety of problems/data types.

Since the original conference version of this paper was published, several industrial and academic groups have contacted us to report the utility of CDM on various problems. For example, Christen and Goiser report success in using CDM for data linkage and reduplication problems (Christen and Goiser 2005). They consider whether our simpler formulation of compression based distance loses any utility compare to the complicated (to implement) original formulation (Li et al. 2003) and note "*in most cases the CDM performs slightly better than the NCD*". Mahoney and Chan of Interface & Control Systems, Inc. tested the anomaly detection algorithm introduced in Sect. 3.2 on solenoid current readings from Marotta series fuel control valves used on the Space Shuttle (TEK dataset), and they "*…confirmed its ability to detect anomalies in the TEK series traces*" (Mahoney and Chan 2005).

There are many directions in which this work may be extended. We intend to perform a more rigorous theoretical analysis of the CDM measure. For example, CDM is a dissimilarity measure; if it could be modified to be a *distance measure*, or better still, a *distance metric*, we could avail of a wealth of pruning and indexing techniques to speed up classification Ratanamahatana and Keogh (2004), clustering Elkan (2003), and similarity search Vlachos et al. (2003). While it is unlikely that CDM can be transformed in a true metric, it may be possible to prove a weaker version of the triangular inequality, which can be bounded and used to prune the search space (Elkan 2003). A step in this direction is demonstrated in the recent paper (Sculley and Brodley 2006).

Finally, we note that our approach is clearly not suitable for classifying or clustering low dimensionality data (although Fig. 2 shows exceptionally good results on time series with only 1,000 data points). We plan to theoretically and empirically investigate the limitations on object sizes that we can meaningfully work with using our proposed approach.

Acknowledgements Thanks to Paul Vitanyi for useful comments and suggestions. Thanks to Ming Li for his feedback on Sect. 4.1, and to the many donors of datasets, especially Manuela Veloso and Douglas Vail at Carnegie Mellon University for the robot data. Thanks also to Stephen Bay for his many useful comments. We particularly thank the reviewers for their useful and constructive suggestions. All datasets used in this paper are available for free download from (Keogh 2004). For convenience, we also include the Yahoo dataset; however, the copyright remains the property of Yahoo! Inc.

A preliminary version of this paper was presented at the 2004 *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Seattle, WA, and included in its *Proceedings*.

This work is supported in part by grants NSF CAREER IIS-0447773, and NSF DBI-0321756.

Appendix

A table of all time series distance/dissimilarity/similarity measures discussed in Sect. 4.1.1

Euclidean	Manhattan	Supremum	Mahalanobis	Correlation
LPC cepstra	Autocorrelation	Aligned subsequence (Keogh and Ka setty 2002)	Piecewise normalization (Keogh and Ka setty 2002)	Cepstrum (Keogh and Ka setty 2002)
String (suffix tree) (Keogh and Ka setty 2002)	Important points (Keogh and Ka setty 2002)	Edit distance (Keogh and Ka setty 2002)	String signature (Keogh and Ka setty 2002)	Cosine wavelets (Keogh and Ka setty 2002)
Piecewise probabilistic (Keogh and Ka setty 2002)	LCSS (Vlachos et al. 2003)	Bounding rectangles	Shape description language	Multiscale matching
Important points (pratt & fink)	Important points (Fu, Chung, Ng and Luk)	Qualitative similarity index	Feature Base approach (alcock)	ARIMA
Rules (Geurts)	Rules (kadous)	HMM (Smyth & Ge) (Li and biswas)	HMM	Segment-based-warping
Symbolic (MALM)	Symbolic (AIM)	Symbolic (IMPACTS)	Symbolic Bozkaya et. al	Symbolic (Signature Files)
Atomic matching	Vector quantization	Landmarks	Random mapping	Wavelet correlation
DTW-Supremum	Chernoff information divergence	Simple Linear Regression (Lei et al.)	Itakura Measure	Gavrilov et al. Euclidean variants (Domingos 1998)
Kullback–Liebler				

References

Allison L, Stern L, Edgoose T, Dix TI (2000) Sequence complexity for biological sequence analysis. Comput Chem 24(1):43–55

Baronchelli A, Caglioti E, Loreto V (2005) Artificial sequences and complexity measures. J. Stat. Mech: Theory and Exp, Issue 04, P04002

Benedetto D, Caglioti E, Loreto V (2002) Language trees and zipping. Phys Rev Lett 88: 048702

Chakrabarti D, Papadimitriou S, Modha D, Faloutsos C (2004) Fully automatic cross-assocations, In: Proceedings of the KDD 2004, Seattle, WA

Christen P, Goiser K (2005) Towards automated data linkage and deduplication. Tech Report. Australian National University

Cook D, Holder LB (2000) Graph-based data mining. IEEE Intell Syst 15(2):32–41

Dasgupta D, Forrest S (1999) Novelty detection in time series data using ideas from immunology. In: Proc. of the international conference on intelligent systems, Heidelberg, Germany

Domingos P (1998) A process-oriented heuristic for model selection. In: Machine learning Proc. of the fifteenth international conference,. Morgan Kaufmann Publishers, San Francisco, CA, pp 27–135

Elkan, C (2001) Magical thinking in data mining: lessons from CoIL challenge 2000. In Proc. of SIGKDD 2001, San Francisco, CA, USA, pp 426–431

Elkan C (2003) Using the triangle inequality to accelerate k-Means. In: Proc. of ICML 2003, Washington DC, USA, pp 147–153

Faloutsos C, Lin K (1995) FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Proc. of 24th ACM SIGMOD, San Jose, CA, USA

Farach M, Noordewier M, Savari S, Shepp L, Wyner A, Ziv J (1995) On the entropy of DNA: algorithms and measurements based on memory and rapid convergence. In: Proc. of the symp. on discrete algorithms, San Francisco, CA, USA pp 48-57

Ferrandina F, Meyer T, Zicari R (1994) Implementing lazy database updates for an object database system. In: Proc. of the 20 international conference on very large databases, Santiago de Chile, Chile, pp 261–272

Flexer A (1996) Statistical evaluation of neural networks experiments: minimum requirements and current practice. In: Proc. of the 13th european meeting on cybernetics and systems research, vol. 2, Austria, pp 1005–1008

Frank E, Chui C, Witten I (2000) Text categorization using compression models. In: Proc. of the IEEE data compression conference, Snowbird, Utah, IEEE Comput Soc p555

Gaussier E, Goutte C, Popat K, Chen F (2002) A hierarchical model for clustering and categorising documents source lecture notes in computer science; Vol. 2291 archive Proceedings of the 24th BCS-IRSG european colloquium on IR research: advances in information retrieval, Glasgow, UK

Gatlin L (1972) Information theory and the living systems. Columbia University Press, columbia

Gavrilov M, Anguelov D, Indyk P, Motwahl R (2000) Mining the stock market: which measure is best? In: Proc. of the 6th ACM SIGKDD, 2000, Boston, MA, USA

Ge X, Smyth P (2000) Deformable Markov model templates for time-series pattern matching. In: Proc. of the 6th ACM SIGKDD, Boston, MA, pp 81–90

Goldberger A.L, Amaral L, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE (2000) PhysioBank, physioToolkit, and physioNet: components of a new research resource for complex physiologic signals. Circulation 101(23):e215–e220

Kalpakis K, Gada D, Puttagunta V (2001) Distance measures for effective clustering of ARIMA time-series. In: Proceedings of the 1st IEEE ICDM, San Jose, CA, pp 273-280

Kennel M (2004) Testing time symmetry in time series using data compression dictionaries. Phys Rev E 69; 056208

Keogh E. http://www.cs.ucr.edu/∼eamonn/SIGKDD2004, University of California, Riverside

Keogh E, Folias T (2002) The UCR time series data mining archive. University of California, Riverside CA [http://www.cs.ucr.edu/∼eamonn/TSDMA/index.html]

Keogh E, Kasetty S (2002) On the need for time series data mining benchmarks: a survey and empirical demonstration. In: Proc. of SIGKDD, Edmonton, Alberta, Canada

Keogh E, Lin J, Truppel W (2003) Clustering of time series subsequences is meaningless: implications for past and future research. In: Proc. of the 3rd IEEE ICDM, Melbourne, FL, pp 115–122

Kit C (1998) A goodness measure for phrase learning via compression with the MDL principle. In: Kruijff-Korbayova I (ed) The ELLSSI-98 student session, Chapt 13, Saarbrueken, pp 175–187
Li M, Badger JH, Chen X, Kwong S, Kearney P, Zhang H (2001) An information-based sequence distance and its application to whole mitochondrial genome phylogeny. Bioinformatics 17:149–154
Li M, Chen X, Li X, Ma B, Vitanyi, P (2003) The similarity metric. In: Proc. of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, Baltimore, MD, USA, pp 863–872
Li M, Vitanyi P (1997) An introduction to kolmogorov complexity and its applications, 2nd edn, Springer Verlag, Berlin
Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. In: Proc. of the 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery, San Diego, CA
Loewenstern D, Hirsh H, Yianilos P, Noordewier M (1995) DNA sequence classification using compression-based induction, DIMACS Technical Report 95-04
Loewenstern D, Yianilos PN (1999) Significantly lower entropy estimates for natural DNA sequences, J Comput Biol 6(1)
Ma J, Perkins S (2003) Online novelty detection on temporal sequences. In: Proc. international conference on knowledge discovery and data mining, Washington, DC
Mahoney M, Chan P (2005) Learning rules for time series anomaly detection. SensorMiner Tech report (available at [www.interfacecontrol.com/products/sensorMiner/])
Mehta M, Rissanen J, Agrawal R (1995) MDL-based decision tree pruning, In: Proceedings of the first international conference on knowledge discovery and data mining (KDD'95), Montreal, Canada
Needham S, Dowe D(2001) Message length as an effective ockham's razor in decision tree induction, In: Proc. 8th international workshop on AI and statistics, Key West, FL, USA, pp 253–260
Ortega A, Beferull-Lozano B, Srinivasamurthy N, Xie H (2000) Compression for recognition and content based retrieval. In: Proc. of the European signal processing conference, EUSIPCO'00, Tampere, Finland
Papadimitriou S, Gionis A, Tsaparas P, Väisänen A, Mannila H, Faloutsos C (2005) Parameter-free spatial data mining using MDL, In: Proc of the 5th International Conference on Data Mining (ICDM), Houston, TX, USA
Quinlan JR, Rivest RL (1989) Inferring decision trees using the minimum description length principle. Infor Comput 80:227–248
Ratanamahatana CA, Keogh E (2004) Making time-series classification more accurate using learned constraints. In: Proc. of SIAM international conference on data mining (SDM '04), Lake Buena Vista, Florida
Rissanen J (1978) Modeling by shortest data description. Automatica, vol 14:465–471
Salzberg SL (1997) On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Min Knowl Disc 1(3):317–328
Segen J (1990) Graph clustering and model learning by data compression. In: Proc. of the machine learning conference, Austin, TX, USA, pp 93–101
Sculley D, Brodley CE (2006) Compression and machine learning: a new perspective on feature space vectors, In: Proceedings of data compression conference, Snowbird, UT, USA, pp 332–341
Shahabi C, Tian X, Zhao W (2000) TSA-tree: a wavelet-based approach to improve the efficiency of multi-level surprise and trend queries. In: Proc. of the 12th Int'l conference on scientific and statistical database management (SSDBM 2000), Berlin, Germany
Teahan WJ, Wen Y, McNab RJ, Witten IH (2000) A compression-based algorithm for Chinese word segmentation. Comput Linguist 26:375–393
Vlachos M, Hadjieleftheriou M, Gunopulos D, Keogh E (2003) Indexing multi-dimensional time-series with support for multiple distance measures. In: Proc. of the 9th ACM SIGKDD, Washington, DC, USA, pp 216–225
Wallace C, Boulton (1968) An information measure for classification, Comput J 11 (2):185–194
Yairi T, Kato Y, Hori K (2001) Fault detection by mining association rules from house-keeping data. In: Proc. of Int'l sym. on AI, Robotics and Automation in Space

Biographical Sketches

Eamonn Keogh is an assistant professor at the University of California, Riverside. He received his Ph.D. from the University of California, Irvine. His research interests include artificial intelligence, data mining, and information retrieval.

Stefano Lonardi is an assistant professor at the University of California, Riverside. He received his Ph.D. from Purdue University. His research interests include computational biology, data mining, and data compression.

Sang Hee-Lee is an assistant professor at the University of California, Riverside. Her research interests include evolution of human morphological variation with emphasis on sexual dimorphism.

Chotirat Ann Ratanamahatana is a lecturer at Computer Engineering Department, Chulalongkorn University, Bangkok, Thailand. She received her undergraduate and graduate studies from Carnegie Mellon University and Harvard University, respectively, and received her Ph.D. from the University of California, Riverside. Her research interests include data mining, information retrieval, machine learning, and human-computer interaction.

Li Wei is a Ph.D candidate in Department of Computer Science & Engineering at the University of California, Riverside. She received her B.S. and M.S. degrees from Fudan University, China. Her research interests include data mining and information retrieval.

John Handley is a member of the research staff at Xerox Corporation. He earned a Ph.D. from Rochester Institute of Technology. Among his research interests are data mining, psychometrics, and nonlinear image processing.